

Wydanie VIII



Rogers Cadenhead

Java

w 24 godziny 

SAMS

Helion 

Tytuł oryginału: Sams Teach Yourself Java in 24 Hours, Eighth Edition

Tłumaczenie: Zbigniew Waśko (wstęp, rozdz. 1 – 7, 10 – 24, dodatki), Paweł Józwiak (rozdz. 8, 9)

ISBN: 978-83-283-4235-4

Authorized translation from the English language edition, entitled: JAVA IN 24 HOURS, SAMS TEACH YOURSELF (COVERING JAVA 9), Eighth Edition; ISBN 0672337940; by Rogers Cadenhead; published by Pearson Education, Inc, publishing as SAMS Publishing. Copyright © 2018 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. Polish language edition published by HELION S.A., Copyright © 2018.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Projekt okładki: Studio Gravite / Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/jav24h.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/jav24h>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

	Dedykacja	11
	O autorze	12
	Podziękowania	12
	Wprowadzenie	13
Godzina 1.	Zostań programistą	15
	Wybór języka	16
	Mówienie komputerowi, co ma robić	18
	Jak działają programy	19
	Gdy program nie chce działać	20
	Wybór narzędzia programistycznego	21
	Instalowanie narzędzia programistycznego	22
	Podsumowanie	22
	Warsztaty	23
Godzina 2.	Napisz swój pierwszy program	27
	Co jest potrzebne do napisania programu?	27
	Przygotowanie do pisania programu Saluton	28
	Rozpoczynamy pisanie programu	28
	Przechowywanie informacji w zmiennych	32
	Zapisywanie kompletnego programu	32
	Kompilowanie programu do pliku klasy	33
	Naprawianie błędów	34
	Uruchamianie programu	35
	Podsumowanie	36
	Warsztaty	37
Godzina 3.	Wycieczka z Javą	41
	Przystanek pierwszy — Oracle	41
	Java w szkole	44
	Przerwa na lunch w kuchni Food Network	45

	Oglądanie nieba w NASA	46
	Powrót do spraw przyziemnych	47
	Przystanek SourceForge	49
	Podsumowanie	50
	Warsztaty	51
Godzina 4.	Jak działają programy pisane w Javie	53
	Tworzenie aplikacji	53
	Przekazywanie argumentów do aplikacji	55
	Biblioteka klas Javy	57
	Testowanie instrukcji Javy w JShell	61
	Podsumowanie	63
	Warsztaty	63
Godzina 5.	Przechowywanie i modyfikowanie informacji	65
	Instrukcje a wyrażenia	65
	Określanie typu zmiennej	66
	Nazywanie zmiennych	70
	Przechowywanie informacji w zmiennych	71
	Wszystko o operatorach	72
	Stosowanie wyrażeń matematycznych	76
	Podsumowanie	78
	Warsztaty	78
Godzina 6.	Komunikacja przy użyciu łańcuchów	81
	Przechowywanie tekstów w formie łańcuchów	81
	Wyświetlanie łańcuchów	82
	Umieszczanie znaków specjalnych w łańcuchach	83
	Sklejanie łańcuchów	84
	Wstawianie zmiennych do łańcucha	84
	Zaawansowana obsługa łańcuchów	85
	Wyświetlanie napisów końcowych	88
	Podsumowanie	89
	Warsztaty	90
Godzina 7.	Instrukcje warunkowe a podejmowanie decyzji	93
	Instrukcja if	94
	Instrukcja if-else	97
	Instrukcja switch	97

Operator warunkowy	99
Wyświetlanie zegara	100
Podsumowanie	104
Warsztaty	105
Godzina 8. Powtarzanie działań za pomocą pętli	109
Pętle for	109
Pętle while	112
Pętle do-while	113
Zakończenie działania pętli	114
Nazwanie pętli	115
Testowanie szybkości komputera	116
Podsumowanie	118
Warsztaty	118
Godzina 9. Przechowywanie informacji w tablicach	121
Tworzenie tablic	122
Używanie tablic	123
Tablice wielowymiarowe	125
Sortowanie tablic	126
Liczenie znaków w łańcuchach	128
Podsumowanie	130
Warsztaty	131
Godzina 10. Utwórz swój pierwszy obiekt	133
Na czym polega programowanie obiektowe?	133
Obiekty w akcji	135
Czym są obiekty?	136
Dziedziczenie	137
Budowanie hierarchii dziedziczenia	138
Przekształcanie obiektów i prostych zmiennych	139
Tworzenie obiektu	143
Podsumowanie	146
Warsztaty	146
Godzina 11. Nadawanie cech obiektowi	149
Tworzenie zmiennych	149
Tworzenie zmiennych klasowych	151
Definiowanie zachowań klas	152

	Zagnieżdżanie klas	157
	Stosowanie słowa kluczowego this	159
	Stosowanie metod i zmiennych klasowych	160
	Podsumowanie	162
	Warsztaty	162
Godzina 12.	Wykorzystuj maksymalnie obiekty istniejące	165
	Moc dziedziczenia	165
	Ustanawianie dziedziczenia	167
	Wykorzystywanie obiektów istniejących	169
	Przechowywanie obiektów tej samej klasy na listach tablicowych	170
	Tworzenie podklasy	174
	Podsumowanie	176
	Warsztaty	177
Godzina 13.	Przechowywanie obiektów w strukturach danych	179
	Lista tablicowa	179
	Mapy	185
	Podsumowanie	189
	Warsztaty	189
Godzina 14.	Obsługa błędów	193
	Wyjątki	194
	Zgłaszanie wyjątków i ich przechwytywanie	203
	Podsumowanie	206
	Warsztaty	206
Godzina 15.	Tworzenie programu z użyciem wątków	209
	Wątki	209
	Stosowanie wątków	214
	Konstruktor	216
	Przechwytywanie błędów przy wprowadzaniu adresów URL	216
	Uruchamianie wątku	217
	Obsługa kliknięć myszą	218
	Cykliczne wyświetlanie łączy	219
	Podsumowanie	222
	Warsztaty	223

Godzina 16.	Stosowanie klas wewnętrznych i domknięć	225
	Klasy wewnętrzne	226
	Domknięcia	232
	Podsumowanie	236
	Warsztaty	237
Godzina 17.	Budowanie prostego interfejsu użytkownika	239
	Biblioteki Swing i AWT	239
	Komponenty interfejsu graficznego	240
	Podsumowanie	257
	Warsztaty	257
Godzina 18.	Projektowanie interfejsu użytkownika	261
	Posługiwanie się menedżerami układu	261
	Projektowanie interfejsu aplikacji	266
	Podsumowanie	271
	Warsztaty	272
Godzina 19.	Reagowanie na działania użytkownika	275
	Implementacja nasłuchu zdarzeń	275
	Ustawianie komponentów, żeby były słyszane	276
	Obsługa zdarzeń wywołanych przez użytkownika	277
	Kompletowanie aplikacji	281
	Podsumowanie	290
	Warsztaty	290
Godzina 20.	Zapisywanie i odczytywanie plików	293
	Strumienie	293
	Zapisywanie danych za pomocą strumienia	301
	Zapisywanie i odczytywanie parametrów konfiguracyjnych	303
	Podsumowanie	306
	Warsztaty	307
Godzina 21.	Korzystanie z nowego klienta HTTP	309
	Moduły Javy	309
	Przygotowanie żądania HTTP	310
	Pobieranie pliku z sieci	314
	Umieszczanie danych w sieci	316
	Podsumowanie	319
	Warsztaty	320

Godzina 22.	Tworzenie grafik przy użyciu biblioteki Java2D	323
	Klasa Font	323
	Klasa Color	324
	Tworzenie kolorów niestandardowych	325
	Rysowanie linii i kształtów	325
	Tworzymy diagram kołowy	329
	Podsumowanie	335
	Warsztaty	336
Godzina 23.	Tworzenie modów do Minecrafta	339
	Konfigurowanie serwera gry Minecraft	340
	Łączenie się z serwerem	344
	Utwórz swój pierwszy mod	346
	Uczymy zombie jeździć na koniach	352
	Wyszukujemy wszystkie moby (i je zabijamy)	358
	Tworzymy mod, który może coś zbudować	362
	Podsumowanie	367
	Warsztaty	367
Godzina 24.	Pisanie aplikacji androidowych	371
	Wprowadzenie do Androida	371
	Tworzenie aplikacji androidowej	373
	Uruchamianie aplikacji	378
	Projektowanie prawdziwej aplikacji mobilnej	380
	Podsumowanie	389
	Warsztaty	390
Dodatek A	Korzystanie ze zintegrowanego środowiska programistycznego NetBeans	393
	Instalacja środowiska NetBeans	393
	Tworzenie nowego projektu	394
	Tworzenie nowej klasy Javy	396
	Uruchamianie aplikacji	398
	Naprawianie błędów	398

Dodatek B	Co dalej: zasoby związane z Javą	401
	Inne ważne uwagi książki	401
	Oficjalna strona internetowa firmy Oracle poświęcona Javie	402
	Inne witryny poświęcone Javie	402
	Spotkania użytkowników Javy	404
	Oferty pracy	404
Dodatek C	Witryna internetowa książki	407
Dodatek D	Rozwiązywanie problemów związanych z emulatorem wbudowanym w Android Studio	409
	Problemy z uruchamianiem aplikacji	409
Dodatek E	Naprawianie błędów związanych z niewidocznymi pakietami w NetBeans	415
	Dodawanie informacji o module	415
	Skorowidz	417

Godzina 2.

Napisz swój pierwszy program

Zagadnienia do nauki:

- ▶ wpisywanie programu w edytorze tekstu,
- ▶ porządkowanie programu za pomocą nawiasów,
- ▶ zapisywanie informacji w zmiennej,
- ▶ wyświetlanie informacji przechowywanej w zmiennej,
- ▶ zapisywanie, kompilowanie i uruchamianie programu.

Jak już pisałem w godzinie 1., „Zostań programistą”, program komputerowy jest zestawem instrukcji, które mówią komputerowi, co ma robić. Instrukcje te są przekazywane komputerowi przy użyciu języka programowania.

W ciągu tej godziny utworzysz swój pierwszy program w Javie, wpisując instrukcje w edytorze tekstu. Następnie zapiszesz program, skompilujesz go i przetestujesz. Potem spróbujesz go celowo zepsuć i naprawić, tylko po to, żeby zobaczyć, jak to się robi.

Co jest potrzebne do napisania programu?

Jak już wiadomo z godziny 1., do tworzenia programów w Javie potrzebne jest narzędzie programistyczne obsługujące pakiet Java Development Kit (JDK), jakim jest np. zintegrowane środowisko programistyczne (IDE) NetBeans. Potrzebujesz narzędzia, które skompiluje i uruchomi programy napisane w Javie, oraz edytor tekstowy, w którym napiszesz te programy.

W większości przypadków programy komputerowe pisze się w edytorze tekstu (zwanym także edytorem kodu źródłowego). Niektóre języki programowania mają własne edytory. Środowisko NetBeans zawiera taki edytor do pisania programów w Javie.

Programy pisane w Javie są zwykłymi plikami tekstowymi bez specjalnego formatowania, np. wyśrodkowania lub pogrubienia. Edytor kodu źródłowego w NetBeans działa jak prosty edytor tekstu, ale jest wyposażony w kilka usprawnień przydatnych dla programistów. Podczas wprowadzania tekstu zmienia się jego kolor zgodnie z poszczególnymi elementami języka. NetBeans tworzy także odpowiednie wcięcia akapitów i zapewnia pomocną dokumentację programistyczną.

Ponieważ programy Javy są plikami tekstowymi, można je otwierać i edytować w każdym edytorze tekstu. Możesz napisać program w środowisku NetBeans, otworzyć go w windowsowym Notatniku, wprowadzić tam zmiany i ponownie, bez żadnych problemów, otworzyć w edytorze NetBeans.

Przygotowanie do pisania programu Saluton

Pierwszy program, który napiszesz, wyświetla tradycyjne w świecie informatyków pozdrowienie: „Saluton mondo!” (czyli „Witaj, świecie”).

Jeśli jeszcze tego nie zrobiłeś, przygotuj pierwszy projekt programistyczny w NetBeans. W tym celu utwórz nowy projekt o nazwie *Java24*, wykonując następujące czynności:

1. Z menu *File* wybierz polecenie *New Project*. Otworzy się okno dialogowe *New Project*.
2. Wybierz kategorię projektu *Java* oraz typ projektu *Java Application*, a następnie kliknij przycisk *Next*.
3. Jako nazwę projektu wpisz **Java24** (jeśli wcześniej utworzyłeś już projekt o tej nazwie, pojawi się komunikat o błędzie informujący, że taki folder już istnieje i nie jest pusty — *Project folder already exists and is not empty*).
4. Wyłącz tworzenie klasy głównej przez usunięcie zaznaczenia pola wyboru *Create Main Class*.
5. Kliknij przycisk *Finish*, aby zakończyć tworzenie nowego projektu.

Dla projektu *Java24* został utworzony nowy folder. Z tego projektu możesz korzystać przy pisaniu wszystkich programów Javy omawianych w tej książce.

Rozpoczynamy pisanie programu

NetBeans grupuje powiązane programy we wspólny projekt. Jeśli projekt *Java24* nie jest otwarty, wykonaj poniższe czynności:

1. Wybierz polecenie *File/Open Project*. Pojawi się okno dialogowe wyboru pliku.
2. Odszukaj i wskaż folder *NetBeansProjects* (jeśli nie jest jeszcze wybrany).
3. Zaznacz pozycję *Java24* i kliknij przycisk *Open Project*.

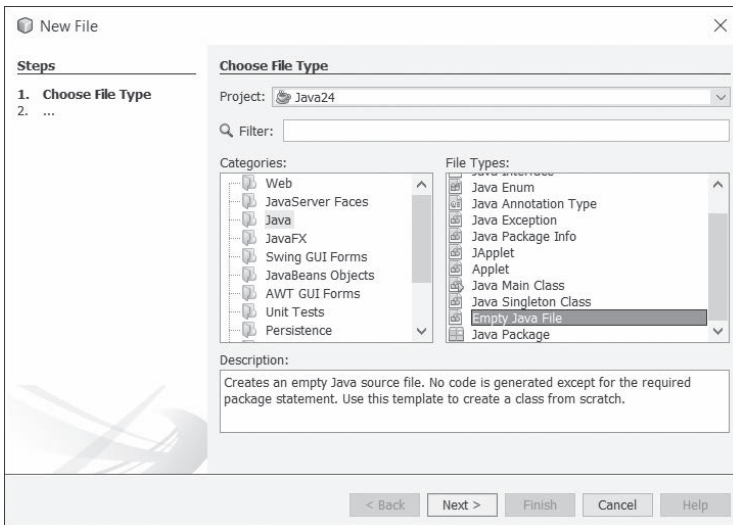
Projekt *Java24* pojawi się na panelu *Projects* obok ikony filiżanki kawy oraz znaku plus (+), którego kliknięcie rozwija listę plików i folderów wchodzących w skład projektu.

Aby dodać nowy program do aktualnie otwartego projektu, wybierz polecenie *File/New File*. Otworzy się kreator *New File* pokazany na rysunku 2.1.

W panelu *Categories* wyświetlane są różne rodzaje programów, które możesz utworzyć. Kliknij pozycję *Java*, aby zobaczyć, jakie typy plików należą do tej kategorii. Na potrzeby swojego pierwszego projektu wybierz typ *Empty Java File* (na końcu listy *File Types*) i kliknij przycisk *Next*.

Otworzy się okno dialogowe *New Empty Java File*. Wykonaj poniższe czynności, aby rozpocząć pisanie programu:

1. W polu tekstowym *Class Name* wpisz nazwę klasy **Saluton**.
2. W polu *Package* wpisz nazwę pakietu **com.java24hours**.
3. Kliknij przycisk *Finish*.



RYSUNEK 2.1. Kreator nowego pliku

Możesz już teraz rozpocząć pracę nad swoim programem — w edytorze kodu źródłowego otworzy się pusty plik o nazwie *Saluton.java*. Korzystając z tego edytora, rozpocznij karierę programisty i wprowadź wszystkie wiersze z listingu 2.1. Zawarte tam instrukcje noszą nazwę kodu źródłowego programu.

LISTING 2.1. Program Saluton

```

1: package com.java24hours;
2:
3: class Saluton {
4:     public static void main(String[] arguments) {
5:         // Tutaj trafi mój pierwszy program w Javie.
6:     }
7: }
```

Ostrzeżenie

Nie wprowadzaj numeru wiersza i dwukropka na początku poszczególnych wierszy — są one dodawane jedynie po to, aby ułatwić odnoszenie się do poszczególnych fragmentów kodu.

Pamiętaj, aby przepisać wszystko dokładnie tak, jak jest w listingu (włącznie z wielkością liter), i za pomocą spacji lub klawisza *Tab* wprowadź wcięcia w wierszach od 4. do 6. Gdy skończysz, wybierz polecenie *File/Save*, aby zapisać plik.

W tym momencie plik *Saluton.java* zawiera jedynie podstawy programu Java. Napiszesz wiele programów, które będą wyglądać tak jak ten, z wyjątkiem słowa *Saluton* w wierszu 3.

Stanowi ono nazwę programu i za każdym razem będzie inne. Zawartość wiersza 5. na pewno rozumiesz, ponieważ jest napisany w języku, którego używasz na co dzień. Reszta może być nowością.

Instrukcja `class`

Pierwszy wiersz programu ma następującą postać:

```
package com.java24hours;
```

Pakiet (package) to w Javie sposób na grupowanie programów. Ten wiersz informuje komputer, aby nadał pakietowi nazwę `com.java24hours`.

Po pustym wierszu znajduje się trzeci wiersz o treści:

```
class Saluton {
```

W przykładzie na język codzienny oznacza to: „Komputerze, nadaj mojemu programowi nazwę `Saluton`”.

Jak pewnie przypominasz sobie z godziny 1., każde polecenie wydawane komputerowi jest nazywane instrukcją. Za pomocą instrukcji `class` nadajesz programowi komputerowemu nazwę. Służy ono także do zdefiniowania innych danych programu, o których dowiesz się nieco później. Pojęcie klasy (`class`) ma istotne znaczenie, ponieważ programy pisane w Javie także są nazywane klasami.

W prezentowanym przykładzie nazwa programu `Saluton` odpowiada nazwie pliku *Saluton.java*. Program w Javie musi mieć nazwę pasującą do pierwszej części nazwy jego pliku, czyli tej, która znajduje się przed znakiem kropki (`.`). Obie nazwy powinny być także napisane literami o takiej samej wielkości.

Jeśli nazwa programu nie odpowiada nazwie pliku, próba skompilowania niektórych programów może zakończyć się komunikatem o błędzie — wszystko zależy od tego, w jaki sposób instrukcja `class` została użyta do skonfigurowania programu.

Za co odpowiada instrukcja `main`?

Kolejny wiersz programu ma następującą postać:

```
public static void main(String[] arguments) {
```

Mówi on komputerowi: „Tutaj zaczyna się główna część programu”. Programy w Javie składają się z różnych sekcji. Musi więc istnieć sposób na zidentyfikowanie tej części programu, która po jego uruchomieniu jest wykonywana jako pierwsza.

Instrukcja `main` jest punktem początkowym większości programów pisanych w Javie. Wyjątki stanowią aplety, czyli programy uruchamiane na stronie internetowej przez przeglądarkę, serwlety, czyli programy uruchamiane przez serwer sieciowy, oraz aplikacje mobilne, czyli programy pracujące na urządzeniach mobilnych.

Większość programów opisanych w tej książce używa instrukcji `main` jako punktu początkowego. Wynika to z faktu, że będą uruchamiane na komputerze. Aplety i serwlety są uruchamiane pośrednio przez inny program lub urządzenie.

Dla odróżnienia programy uruchamiane bezpośrednio przez użytkownika są nazywane aplikacjami.

Nawiasy klamrowe

W programie `Saluton` wiersze 3., 4., 6. i 7. zawierają znaki nawiasów klamrowych `{` oraz `}`. Służą one do grupowania wierszy programu (podobnie jak zwykle nawiasy grupują słowa w zdaniu). Wszystko, co znajduje się między nawiasem otwierającym `{` a zamykającym `}`, należy do tej samej grupy.

Takie zgrupowania noszą nazwę bloków. W listingu 2.1 nawias otwierający w wierszu 3. jest związany z nawiasem zamykającym w wierszu 7., tworząc w ten sposób blok stanowiący cały program. Takie stosowanie nawiasów pozwala wskazać początek i koniec programu.

Bloki mogą znajdować się także wewnątrz innych bloków (podobnie jak nawiasy stosowane w tym zdaniu (w którym użyty jest też drugi zestaw nawiasów)). W wierszach 4. i 6. programu `Saluton` znajdują się nawiasy definiujące kolejny blok. Blok ten zaczyna się od instrukcji `main`. Wiersze należące do bloku `main` będą wykonywane jako pierwsze po uruchomieniu programu.

Wskazówka

NetBeans pomaga określić miejsca, w których dany blok się rozpoczyna i kończy. Kliknij jeden z nawiasów klamrowych w kodzie źródłowym programu `Saluton`. Nawias ten podświetli się na żółto wraz z odpowiadającym mu drugim nawiasem. Instrukcje Javy zamknięte między tymi dwoma nawiasami tworzą blok. Ta wskazówka nie jest szczególnie przydatna w przypadku tak krótkiego programu jak `Saluton`, ale przy pisaniu dłuższych kodów znacznie ułatwi Ci pracę.

Wewnątrz tego bloku znajduje się jedynie poniższa informacja:

```
// Tutaj trafi mój pierwszy program w Javie.
```

Wiersz ten jest tzw. wypełniaczem. Dwa znaki ukośnika (`//`) na początku wiersza mówią komputerowi, aby go zignorował, ponieważ wiersz został umieszczony jedynie dla osób analizujących kod źródłowy. Wiersze służące do takiego celu określa się jako *komentarze*.

Tak oto napisałeś kompletny program w Javie. Możesz go skompilować, ale po uruchomieniu nic się nie stanie. Nie powiedziałaś bowiem jeszcze komputerowi, co ma robić. Blok instrukcji `main` zawiera jedynie komentarz, który jest ignorowany przez komputer. Musisz wstawić jakieś instrukcje między nawiasami otwierającym i zamykającym bloku `main`.

Przechowywanie informacji w zmiennych

W swoich programach będziesz potrzebował miejsca, w którym byłoby można przechować informację przez pewien czas. Służy do tego zmienna, czyli przestrzeń magazynowa przechowująca informacje, takie jak liczby całkowite, liczby zmiennoprzecinkowe, wartości prawda/fałsz, znaki i łańcuchy tekstowe. Informacja zapisana w zmiennej może się zmieniać — i właśnie dlatego ten element nosi nazwę zmiennej.

W pliku *Saluton.java* zastąp wiersz 5. następującym:

```
String greeting = "Saluton mondo!";
```

Ta instrukcja mówi komputerowi, aby zapisał tekst „Saluton mondo!” w zmiennej `greeting`.

W programie pisany przy użyciu Javy musisz powiedzieć komputerowi, jaki rodzaj informacji ma być przechowywany w zmiennej. W tym programie zmienna `greeting` jest typu łańcuchowego — może przechowywać ciągi znaków, takich jak litery, cyfry czy znaki interpunkcyjne. Wprowadzone do instrukcji słowo `String` definiuje zmienną jako zdolną do przechowywania łańcuchów.

Przy wprowadzaniu tej instrukcji do programu należy ją zakończyć średnikiem. Średniki kończą wszystkie instrukcje programów pisanych w Javie. Są czymś w rodzaju kropki na końcu zdania. Na ich podstawie komputer może określić, gdzie kończy się jedna instrukcja i rozpoczyna następną.

Wprowadzanie każdej instrukcji w odrębnym wierszu sprawia, że program jest bardziej zrozumiały (dla nas, ludzi).

Wyświetlanie zawartości zmiennej

Jeśli teraz uruchomisz program, nadal będzie się wydawało, że nic się nie dzieje. Polecenie zapisania tekstu w zmiennej `greeting` działa niejako w tle. Jeśli chcesz się przekonać, że komputer jednak coś zrobił, możesz wyświetlić zawartość zmiennej.

Wstaw kolejny pusty wiersz w programie *Saluton*, pod instrukcją `String greeting = "Saluton mondo!"`. W nowym wierszu wpisz następującą instrukcję:

```
System.out.println(greeting);
```

Ta instrukcja mówi komputerowi, aby wyświetlił wartość przechowywaną w zmiennej `greeting`. Instrukcja `System.out.println` każe komputerowi wyświetlić informację na systemowym urządzeniu wyjściowym — na monitorze.

To jest już coś.

Zapisywanie kompletnego programu

Twój program powinien teraz przypominać ten z listingu 2.2, chociaż może mieć nieco inne odstępstwa w wierszach 5. i 6. Wykonaj potrzebne poprawki i zapisz plik (za pomocą polecenia *File/Save*).

LISTING 2.2. Gotowa wersja programu Saluton

```
1: package com.java24hours;
2:
3: class Saluton {
4:     public static void main(String[] arguments) {
5:         String greeting = "Saluton mondo!";
6:         System.out.println(greeting);
7:     }
8: }
```

Gdy komputer uruchomi ten program, wykona poszczególne instrukcje w ramach bloku `main` umieszczone w wierszach 5. i 6. W listingu 2.3 pokazano, jak wyglądałby program, gdyby został napisany w języku polskim, a nie w Javie.

LISTING 2.3. Opis poszczególnych wierszy programu Saluton

```
1: Umieść ten program w pakiecie com.java24hours.
2:
3: Tutaj rozpoczyna się program Saluton:
4:     Tutaj rozpoczyna się główna część programu:
5:         Zapisz tekst „Saluton mondo!” w zmiennej łańcuchowej o nazwie greeting.
6:         Wyświetl zawartość zmiennej greeting.
7:     Tutaj kończy się główna część programu.
8: Tutaj kończy się program Saluton.
```

Listing 2.4 pokazuje, jak wyglądałby ten program, gdyby był napisany w języku klingońskim, czyli języku wojowników z serialu *Star Trek*.

LISTING 2.4. Program Saluton w języku klingońskim

```
1: Ten program należy do rodu com.java2hours!
2:
3: Jeśli nie chcesz, by spotkało Cię coś złego, rozpocznij teraz program Saluton!
4:     Tutaj uroczyście rozpoczyna się główna część programu!
5:         Zapisz bełkot „Saluton mondo!” w zmiennej łańcuchowej o nazwie greeting.
6:         Wyświetl ten bełkot w języku klingońskim!
7:     Aby nie narazić się na mój gniew, zakończ główną część programu!
8: Zakończ teraz program Saluton i ciesz się, że uszedłeś z życiem!
```

Kompilowanie programu do pliku klasy

Zanim uruchomisz program napisany w Javie, musisz go skompilować. Podczas kompilacji instrukcje wydawane w ramach programu są przekształcane do postaci zrozumiałej dla komputera.

NetBeans kompiluje programy automatycznie podczas ich zapisywania. Jeśli udało Ci się wprowadzić wszystko tak jak na listingu 2.2, program skompiluje się prawidłowo.

Zostanie utworzona skompilowana wersja programu, czyli nowy plik o nazwie *Saluton.class*. Wszystkie programy pisane w Javie są kompilowane do plików klasy, które otrzymują rozszerzenie *.class*. Program w Javie może się składać z wielu współpracujących ze sobą klas, ale w tak prostym programie jak *Saluton* potrzebna jest tylko jedna.

Kompilator przekształca kod źródłowy Javy na kod bajtowy, czyli nadaje mu postać, która może być wykonywana przez wirtualną maszynę Javy (JVM).

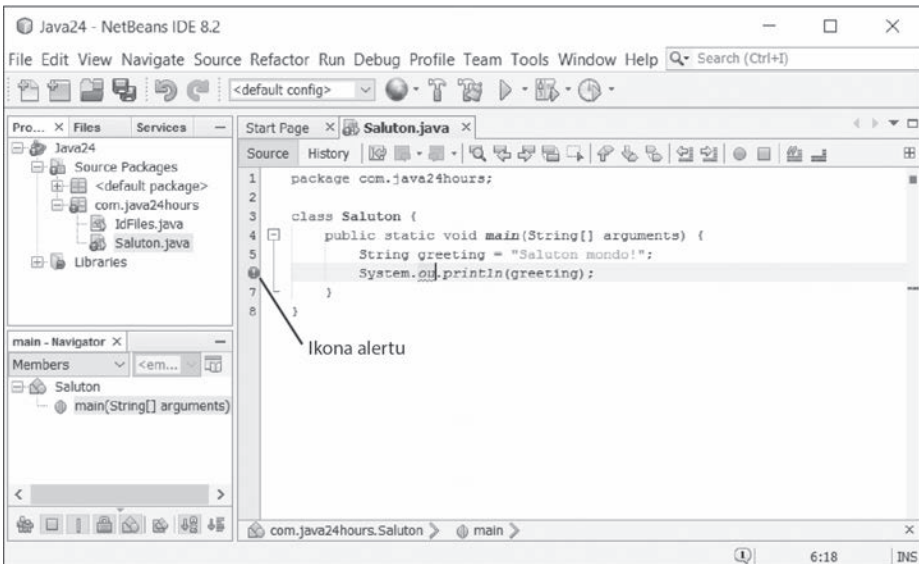
Uwaga

Kompilator Javy odzywa się tylko wtedy, gdy wykryje jakiś błąd, o którym musi poinformować. Jeśli kompilacja przebiegnie pomyślnie bez żadnych problemów, nie wygeneruje żadnej odpowiedzi.

Jest to trochę frustrujące. Kiedy zaczynałem swoją przygodę z Javą, miałem nadzieję, że udanej kompilacji będą towarzyszyć fanfary.

Naprawianie błędów

Podczas pisania programu w edytorze kodu źródłowego, będącym częścią środowiska NetBeans, błędy są oznaczane czerwoną ikoną alertu wyświetlaną przy lewej krawędzi panelu edytora, jak pokazano na rysunku 2.2.



RYСУNEK 2.2. Sygnalizacja błędów w edytorze kodu źródłowego

Ikona wyświetla się w wierszu, w którym błąd został wykryty. Możesz ją kliknąć, aby wyświetlić okno dialogowe błędów zawierające następujące informacje:

- ▶ nazwa programu w Javie,
- ▶ typ błędu,
- ▶ wiersz, w którym wykryto błąd.

Oto przykład zawartości okna błędów, które może się wyświetlić przy kompilowaniu programu `Saluton`:

```
cannot find symbol.  
symbol   : variable greting  
location: class Saluton
```

Komunikatem o błędzie jest pierwszy wiersz w oknie: `cannot find symbol` (nie można odnaleźć symbolu). Te komunikaty często mogą być mylące dla początkujących programistów. Jeśli komunikat o błędzie nie ma według Ciebie sensu, nie trać zbyt wiele czasu, aby go rozszyfrować. Lepiej przyrzeć się problematycznemu wierszowi i sprawdzić go pod kątem najbardziej oczywistych przyczyn.

Przykładowo, czy potrafisz stwierdzić, co jest nie tak w poniższej instrukcji?

```
System.out.println(greting);
```

Błąd polega na literówce w nazwie zmiennej, która powinna mieć postać `greeting` zamiast `greting`. Wprowadź tę literówkę celowo w edytorze NetBeans i zobacz, co się stanie.

Jeśli podczas pisania programu `Saluton` pojawią się okna z błędami, ponownie sprawdź, czy kod odpowiada listingowi 2.2 i popraw wykryte nieścisłości. Upewnij się, że wielkość liter jest prawidłowa i że wstawiłeś wszystkie znaki interpunkcyjne, np. `{`, `}` oraz `;`.

Bardzo często krótkie spojrzenie na wiersz wskazany w oknie błędów wystarczy do znalezienia błędów (lub błędów) i naprawienia go.

Wskazówka

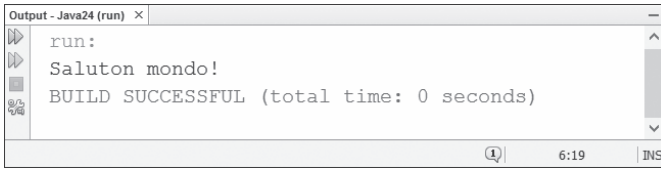
Wskazówka

Na serwerze FTP wydawnictwa: <ftp://ftp.helion.pl/przyklady/jav24h.zip> znajdują się kody źródłowe wszystkich prezentowanych tu programów.

Uruchamianie programu

Aby sprawdzić, czy program `Saluton` rzeczywiście wykonuje to, co powinien, uruchom go za pomocą interpretera Javy. W NetBeans wybierz polecenie *Run/Run File*. Poniżej edytora kodu źródłowego otworzy się panel *Output*. Jeśli kod nie zawiera błędów, w tym miejscu wyświetli się rezultat działania programu, jak pokazano na rysunku 2.3.

Jeśli wyświetli się tekst „*Saluton mondo!*”, to oznacza, że właśnie udało Ci się napisać pierwszy działający program w Javie! Komputer przywitał się ze światem — jest to tradycja w świecie programistów tak samo ważna, jak napoje energetyczne, koszule z krótkimi rękawami i *League of Legends*.



```
Output - Java24 (run) x
run:
Saluton mondo!
BUILD SUCCESSFUL (total time: 0 seconds)
```

RYSUNEK 2.3. Uruchamianie pierwszego programu napisanego w Javie

Pewnie zastanawiasz się, dlaczego tradycyjnym pozdrowieniem jest właśnie „Saluton mondo!”. Zdanie to oznacza „Witaj, świecie!” w esperanto, sztucznym języku opracowanym przez Ludwika Zamenhofa w 1887 roku w celu ułatwienia komunikacji międzynarodowej.

To pozdrowienie jest tradycyjne tylko dlatego, że właśnie próbuję rozpocząć taką tradycję.

Wskazówka

Na swoich stronach internetowych firma Oracle udostępnia wyczerpującą dokumentację języka Java. Nie jest Ci ona potrzebna przy czytaniu tej książki, ponieważ każdy temat jest tu dokładnie omawiany, ale jeśli będziesz chciał poszerzać swoją wiedzę i pisać własne programy, takie zasoby mogą Ci się przydać.

Całą dokumentację można pobrać, ale wygodniejsze jest wyszukiwanie potrzebnych informacji na stronie internetowej. Najbardziej aktualna dokumentacja języka Java znajduje się pod adresem <https://docs.oracle.com/javase/9/>.

Podsumowanie

Podczas tej godziny miałeś szansę napisać swój pierwszy program w Javie. Dowiedziałeś się, że w celu opracowania takiego programu trzeba wykonać cztery następujące czynności podstawowe:

1. Napisać program w edytorze tekstowym lub przy użyciu narzędzia takiego jak NetBeans.
2. Skompilować program do pliku klasy.
3. Rozkazać wirtualnej maszynie Javy, aby uruchomiła klasę.
4. Powiadomić o tym swoją mamę.

Przy okazji poznałeś kilka podstawowych pojęć programistycznych, takich jak kompilatory, interpretry, bloki, instrukcje i zmienne. Zostaną one dokładniej wyjaśnione w trakcie następnych godzin. Jeśli w ciągu tej godziny udało Ci się napisać działający program Saluton, możesz przejść dalej.

(Czwarty krok nie ma nic wspólnego z programowaniem w Javie. Po prostu mama poprosiła mnie, abym o niej wspomniał w książce).

Warsztaty

Pytania i odpowiedzi

P. Jakie znaczenie ma umieszczenie odpowiedniej liczby odstępów na początku wiersza w programie pisany w Javie?

O. Dla komputera nie ma to żadnego znaczenia. Odstępy są przeznaczone jedynie dla ludzi przeglądających program komputerowy — kompilator Javy ignoruje je zupełnie. Możesz napisać program Saluton bez wcięć wykonywanych przy użyciu klawisza spacji lub *Tab*, a i tak zostanie on pomyślnie skompilowany.

Choć liczba spacji na początku wiersza nie ma znaczenia dla komputera, to jednak warto stosować spójny system odstępów i wcięć w swoich programach. Dlaczego? Ponieważ odpowiednie wcięcia ułatwiają zorientowanie się w strukturze programu oraz szybkie sprawdzenie, do której instrukcji należy dany blok.

Programy, które piszesz, powinny być zrozumiałe dla innych programistów, a także dla Ciebie, gdy po kilku tygodniach lub miesiącach spróbujesz naprawić jakiś błąd lub wprowadzić zmianę. Spójne odstępy i wcięcia są czymś, co nazywany stylem programowania. Dobrzy programiści mają swój styl i go zawsze stosują.

P. Program pisany w języku Java został określony jako klasa i jako grupa klas. Czym jest w rzeczywistości?

O. Jednym i drugim. Proste programy, które napiszesz w Javie w ciągu kilku następnych godzin, są kompilowane do jednego pliku z rozszerzeniem *.class*. Możesz je uruchomić za pomocą wirtualnej maszyny Javy. Programy napisane w Javie mogą jednak składać się także z wielu współpracujących ze sobą klas. Ten temat został szczegółowo opisany w godzinie 10. „Utwórz swój pierwszy obiekt”.

P. Skoro każda instrukcja musi być zakończona średnikiem, to dlaczego wiersz komentarza // Tutaj trafi mój pierwszy program w Javie nie kończy się w taki sposób?

O. Komentarze są całkowicie ignorowane przez kompilator. Jeśli wstawisz znaki // do wiersza programu, kompilator zignoruje wszystko to, co znajduje się na prawo od tych znaków. W poniższym przykładzie pokazano komentarz umieszczony w tym samym wierszu co instrukcja:

```
System.out.println(greeting); // Witaj, świecie!
```

P. Nie mogę znaleźć żadnego błędu w wierszu wskazanym przez kompilator. Co mogę zrobić?

O. Numer wiersza wyświetlany w komunikacie o błędzie nie zawsze oznacza miejsce, w którym należy usunąć błąd. Przyjrzyj się instrukcjom znajdującym się bezpośrednio nad komunikatem o błędzie i sprawdź, czy nie zawierają one literówek lub innych błędów. Błąd z reguły znajduje się w tym samym bloku programu.

P. W jaki sposób mogę odwiedzić Antarktydę?

O. Jeśli nie zamierzasz zostać badaczem naukowym lub pracownikiem obsługi (np. kucharzem, elektrykiem lub lekarzem), możesz być jednym z 10 000 ludzi, którzy co roku turystycznie odwiedzają ten mroźny kontynent.

Wyloty są organizowane z Australii, Nowej Zelandii i Ameryki Południowej, a ich koszt wynosi około 1000 dolarów od osoby.

Kursują także statki rejsowe — takie wycieczki trwają od 10 dni do trzech tygodni, a najdroższe z nich kosztują około 25 000 dolarów. Niektóre z takich rejsów dają możliwość popływania kajakiem pomiędzy pingwinami, wejścia na górę lodową, a nawet przespania się w obozie.

Więcej informacji dla chcących zwiedzić Antarktydę znajduje się na stronie Polar Cruises (www.polarcruises.com).

Brytyjska organizacja Antarctic Survey ma dla takich turystów przydatną radę: „Nie wchodź na lodowce ani duże pola śniegu, jeśli nie masz odpowiedniego przygotowania”.

Quiz

Poniżej znajdują się pytania, które pomogą Ci sprawdzić wiedzę nabytą w trakcie tej godziny.

1. Co tak naprawdę robisz, kompilując program?
 - A) Zapisuję go na dysku.
 - B) Konwertuję do postaci bardziej zrozumiałej dla komputera.
 - C) Dodaję go do swojej kolekcji programów.
2. Czym jest zmienna?
 - A) Czymś, co się kołysze, ale nie przewraca.
 - B) Tekstowym elementem programu ignorowanym przez kompilator.
 - C) Miejscem do przechowywania informacji podczas działania programu.
3. Jak nazywa się proces usuwania błędów?
 - A) Demolowanie.
 - B) Debugowanie.
 - C) Dekomponowanie.

Odpowiedzi

1. B. Kompilowanie programu przekształca plik *.java* w plik *.class* lub zestaw plików *.class*.
2. C. Zmienne są jednym ze sposobów przechowywania informacji; w dalszej części książki dowiesz się o innych możliwościach, takich jak tablice lub stałe. Tym, co się kołysze, ale nie przewraca jest wańka-wstańka, a elementem programu ignorowanym przez kompilator jest komentarz.

- 3. B.** Ponieważ błędy w programie komputerowym są nazywane bugami, proces usuwania tych błędów nazywa się debugowaniem. Niektóre narzędzia programistyczne są wyposażone w debugger, który pomaga eliminować błędy. NetBeans ma jeden z najlepszych debuggerów.

Ćwiczenia

Jeśli chcesz trochę dokładniej zgłębić tematy opisane w trakcie tej godziny, spróbuj wykonać poniższe ćwiczenia:

- ▶ Zdanie „Witaj, świecie!” możesz przetłumaczyć na inne języki, używając Tłumacza Google dostępnego pod adresem <http://translate.google.com>. Napisz program, za pomocą którego komputer przywita się ze światem, np. w języku francuskim, włoskim lub portugalskim.
- ▶ Wróć do programu Saluton i wprowadź do niego jeden lub dwa błędy. Przykładowo, usuń średnik z końca wiersza lub zmień tekst `println` na `println` (wprowadzając cyfrę 1 zamiast litery l). Zapisz program i spróbuj go skompilować, a następnie porównaj komunikaty o błędach z wprowadzonymi błędami.

Rozwiązania ćwiczeń znajdziesz na serwerze FTP wydawnictwa: <ftp://ftp.helion.pl/przyklady/jav24h.zip>.

Skorowidz

A

adnotacja @Override, 168
adres
 strony internetowej, 216
 URL, 311
Android, 15
 emulator, *Patrz:* AVD
 historia, 371
 wersja, 374
Android SDK, 42, 372, 379, 410
Android Software Development Kit,
 Patrz: Android SDK
Android Studio, 372, 373, 381, 390
animacja, 210
antialiasing, 324
apka, *Patrz:* aplikacja mobilna
aplet, 225
aplikacja, *Patrz też:* program
 androidowa, 373
 framework, 374
 argument, 55, 56, 63
 błędy, 398
 ekran, *Patrz:* ekran
 ikona, 381, 382
 kompletowanie, 281
 manifest, 382
 mobilna, 41, 371
 projektowanie, 380, 381
 sieciowa, 41
 tworzenie, 53
 uruchamianie, 378, 388, 398
 zasoby, 381, 382
autoboxing, 141, 187
AVD, 377, 378

B

bean, *Patrz:* ziarno
bezpieczeństwo, 46

biblioteka

 AWT, 240, 258
 klas, 55, 57
 dokumentacja, 57, 58
 Tomcat, 57
 klienta HTTP, 309
 Spigot, 357
 Swing, 215, 229, 240, 258
BIOS, 412, 413
blok, 31
 main, 54
 try-catch, 196, 197, 201, 210
 try-catch-finally, 199
błąd, 34, 56, 193, 398, *Patrz też:* wyjątek
 logiczny, 20
 poważny, 193, 206
 składni, 20
bug, 20

C

character, *Patrz:* znak
Creative Commons, 297
czas, 103

D

dane, 170
 pobieranie, 295
 zapisywanie, 301
data, 103
debugowanie, 20, 157, 377
diagram kołowy, 329
domknięcie, *Patrz:* wyrażenie lambda
dziedziczenie, 137, 138, 165
 hierarchia klas, 138
 wielokrotne, 146
dźwięk, 365

E

edytor, 19, 27, 396
 elipsa, 327
 ekran menu, 383
 enkapsulacja, *Patrz:* hermetyzacja
 etykieta, 246, 247

F

folder, 295
 font, 323
 framework aplikacji, 374
 FreeBalls, 44

G

Google Maps, 380
 Gosling James, 17, 42, 49, 393
 gra

- Celtic Heroes, 42
- JSoko, 49
- Minecraft, *Patrz:* Minecraft sieciowa, 42

 GUI, 210, 215, 239, 240, 323, 382

- schemat wizualno-funkcyjny, 242
- Nimbus, 242

H

HAXM, 410, 411, 412
 hermetyzacja, 154
 Hopper Grace, 20

I

IDE, 21, 393

- Android Studio, 409
- NetBeans, 21, 22, 27, 42, 390
 - instalacja, 393, 394
 - zajętość pamięci, 163

 instrukcja, 18, 78, *Patrz też:* metoda, polecenie

- blokowa, 65, 96
 - try-catch, 196
- break, 98, 105, 114
- case, 98
- class, 30, 136, 150
- extends, 137

- if, 94, 96, 105
- if-else, 97
- import, 215, 415
- main, 30
- Math.sqrt, 54, 55
- new, 122
- pętli, *Patrz:* pętla super, 169
- switch, 97, 98, 105
- System.out.print, 82
- System.out.println, 32, 82, 152
- warunkowa, 93, 99

Integrated Development Environment, *Patrz:* IDE

intencja, 387

interfejs, 146

- ActionListener, 218, 232, 276, 277
- daty i czasu, 104
- ItemListener, 278
- KeyListener, 229, 230, 278, 279
- LivingEntity, 359
- nasłuchowy, 275, 276, 277
- Runnable, 209, 210, 211, 232
- Set, 187
- Spigot API, 340
- użytkownika graficzny, *Patrz:* GUI

interpreter, 19, 20

ISS Detector, 46, 47

iteracja, 111

J

Java, 23, 372

- dokumentacja, 36, 163, 402

- historia, 42

- maszyna wirtualna, *Patrz:* JVM

- nazwa, 43

- wydanie, 43, 44

Java 9, 310

Java Class Library, *Patrz:* biblioteka klas

Java Development Kit, *Patrz:* JDK

Java EE, 51, 393

Java Foundation Classes, 258

Java ME, 393

Java SE, 51, 393

JDK, 21, 42, 393

język

- esperanto, 36

- programowania

- BASIC, 16, 23
- C#, 17
- C++, 17, 23, 43, 49
- interpretowany, 19, 23
- Java, *Patrz*: Java
- kompilowany, 23
- PHP, 16
- Smalltalk, 17, 23
- Visual Basic, 16
- wieloplatformowy, 45
- wybór, 16

JFC, 258

Joy Bill, 42

JShell, 20, 61, 62

- zamykanie, 62

JVM, 20, 34, 45, 343

- zajętość pamięci, 163

K

klasa, 30, 49, 57, 134, 136, 226

- ArrayList, 170, 180, 185,
 - Patrz też*: lista tablicowa
- Arrays, 126
- atrybut, 150
- BorderLayout, 264
- BoxLayout, 265
- Collections, 173
- Color, 323, 324
- Error, 206
- Exception, 194, 196, 201, 206
- File, 294, 295, 301
- FileInputStream, 301
- FileOutputStream, 301
- FileReader, 301
- Files, 314
- FileStore, 255
- FileWriter, 301
- FlowLayout, 245, 262, 263
- Font, 323
- FreeSpacePanel, 252, 253, 254
- główna, 396
- Graphics2D, 325
- GridLayout, 263, 270
- HashMap, 186
- HomePage, 204
- HttpClient, 310, 311
- HttpHeaders, 311
- HttpRequest, 311
- HttpRequest.Builder, 310

- HttpResponse, 311
- IceCreamScoop, 364
- InputStreamReader, 167
- Insets, 265
- IOException, 254
- java.time.LocalDateTime, 103
- java.time.temporalfield.ChronoField, 103
- JavaPlugin, 350
- JButton, 240, 245
- JCheckBox, 248
- JComboBox, 248, 249
- JFrame, 215, 241, 266
- JLabel, 246
- JPanel, 252, 270
- JScrollPane, 251
- JTextArea, 250
- JTextField, 240, 247
- JWindow, 240, 241
- KeyAdapter, 230
- KeyEvent, 279
- KeyViewer, 230
- Line2D.Float, 326
- Location, 350
- Logger, 350
- Map, 187
- module-info, 416
- Object, 140
- Optional, 312
- Paths, 255
- Point, 174, 182
- pomocnicza, 157
- Properties, 303
- publiczna, 158
- Random, 58, 60
- Reader, 167
- Root, 54
- RuntimeException, 202
- StringBuffer, 214
- System, 299
- Thread, 209, 210
- UIManager, 242
- URI, 311
- uruchamialna, 210
- View, 386
- wątkowa, *Patrz*: klasa uruchamialna
- wewnętrzna, 158, 159, 226, 236, 326
 - anonimowa, 228, 229, 230, 231, 234, 236, 237
 - tworzenie, 227
 - zalety, 226

klasa
 World, 350
 zsynchronizowana, 189
 klawiatura, 278
 klucz-wartość, 186, 187
 kod
 bajtowy, 34, 294
 źródłowy, 19, 34
 kolekcja LIFO, 189
 kolor, 323
 niestandardowy, 325
 komentarz, 31, 387
 kompilator, 19, 20, 34
 komponent, 240, 245
 generowanie zdarzenia, 276, 277
 odstępny od krawędzi, 265
 tworzenie, 252
 włączanie, 280
 wyłączanie, 280
 konkatenacja, *Patrz:* łańcuch sklejanie
 konsola, 299
 konstruktor, 155
 bezargumentowy, 155, 156
 nadklasy, 168
 podklasy, 168
 kontener, 240, 241, 244, 245, 323
 menedżer układu, 245, 257, 261, 262,
 263, 264, 265
 wymiary, 261, 263
 kształt, 325, 326, 328

L

liczba pierwsza, 211, 212
 LIFO, 189
 linia, 326
 lista
 opcji, *Patrz:* pole kombi
 tablicowa, 170, 180, 185
 rozmiar, 171
 tworzenie, 180
 wielkość, 180, 181

Ł

łańcuch, 63, 81
 długość, 87
 konwersja na liczbę całkowitą, 143
 porównywanie, 86

przeszukiwanie, 87
 pusty, 85, 90
 sklejanie, 84
 wyświetlanie, 82
 łuk, 328

M

Maklyakov Ivan, 44
 mapa, 186
 maszyna wirtualna
 Androida, *Patrz:* AVD
 Javy, *Patrz:* JVM
 metoda, 152, *Patrz też:* instrukcja, polecenie
 actionPerformed, 218, 277
 add, 171, 180, 181, 244
 addActionListener, 276
 addItemListener, 278
 addKeyListener, 278
 append, 214
 argument, 139
 asString, 311
 browse, 219
 build, 311
 close, 167
 contains, 88, 171, 182
 containsKey, 187
 containsValue, 187
 createNewFile, 295
 createTempFile, 314
 currentTimeMillis, 117
 deklarowanie, 152
 delete, 295
 dostępowa, 154
 encode, 317
 entrySet, 187
 equals, 86, 90, 177
 equalsIgnoreCase, 86
 exists, 295
 firstValue, 311, 312
 fromString, 317
 get, 171, 182, 186, 187
 getActionCommand, 277
 getEntityID, 360
 getFileStore, 255
 getId, 386
 getKeyChar, 279
 getKeyCode, 279
 getKeyText, 279

getLivingEntities, 359
getLocation, 360
getName, 295
getOrDefault, 187
getProperty, 304
getSource, 277
getTotalSpace, 255
getUsableSpace, 255
header, 318
headers, 311
indexOf, 87, 88
isPresent, 312
keyPressed, 229, 278, 279
keyReleased, 229, 278, 279
keyTyped, 229, 230, 278, 279
klasowa, 156, 162
length, 87, 295
list, 304
listFiles, 295
log.info, 348
move, 174
newBuilder, 317, 318
nextInt, 59, 60
onCommand, 348
paint, 290
parseDouble, 364
playSound, 365
POST, 318
przesłanie, 167, 168
put, 186
random, 355
read, 296, 299
remove, 181
renameTo, 295
repaint, 218, 290
resume, 214
run, 211
scoopTerrain, 364
send, 311
sendMessage, 360
setColor, 324
setContentView, 386
setDefaultCloseOperation, 242
setEditable, 272
setEnabled, 280
setFont, 324
setLayout, 245, 252, 266
setLookAndFeel, 242, 244, 271
setText, 290

setValue, 254
setWrapStyleWord, 250
size, 181, 187
sleep, 210
sort, 173
spawn, 349
sqrt, 55
start, 214, 217
stop, 214
substring, 298
suspend, 214
sygnatura, 155
toLowerCase, 87
toString, 177
toUpperCase, 87, 90
translate, 174
zasięg, 157
Międzynarodowa Stacja Kosmiczna,
Patrz: 46, 47
Minecraft, 339
mod, 339, 340
polecenie, 359
tworzenie, 346, 347, 351, 352, 362
wyszukiwanie, 358
serwer, 344
konfigurowanie, 340
uruchamianie, 340, 341
umowa, 341
moduł jdk.incubator.httpclient, 415

N

nadklasa, 138
null, 187, 203, 312

O

obiekt, 48, 57, 82, 134
atrybut, 149, *Patrz:* atrybut
JButton, 245, 276
konwersja, 139, 140, 141
rzutowanie, 140, 141
tworzenie, 136, 143, 155
współdzielenie, 169
Object-Oriented Programming,
Patrz: programowanie obiektowe
obszar tekstowy, 250
okno, 241
okrąg, 327

OOB, *Patrz*: programowanie obiektowe
operator

- >, 232
- dekrementacji, 73, 74, 75
- inkrementacji, 73, 74, 75
- matematyczny, 72
- nierówności, 95
- porównania, 75, 94
- postfiksowy, 73, 74
- prefiksowy, 73, 74
- priorytet, 75
- równości, 95
- strzałkowy, *Patrz*: operator ->
- trójargumentowy, *Patrz*: operator warunkowy
- warunkowy, 99, 100

Oracle, 42, 402

Oracle Technology Network, 402

P

pakiet, 151

- Java Development Kit, *Patrz*: JDK
- java.awt, 182, 215, 245, 258
- java.awt.event, 215, 230, 232, 275
- java.awt.geom, 326
- java.io, 58, 215, 301
- java.net, 215
- java.nio, 314
- java.time, 58, 104
- java.time.temporal, 104
- java.util, 58, 151, 186
- javax.swing, 215, 241, 258, 265
- jdk.incubator.http, 310

panel, 252

pasek przewijania, 251

pętla, 109

- do-while, 109, 113
- for, 109, 110, 181, 360
 - przeglądanie listy tablicowej, 172
 - złożona, 116
- licznik, 110
- nazwa, 115
- nieskończona, 118
- while, 109, 112, 114
- zagnieżdżanie, 115
- zakończenie, 114

plik, 19, 294

- .class, 34
- AndroidManifest.xml, 381, 382
- appicon.png, 381
- colors.xml, 381
- config.dat, 304
- ic_launcher.png, 381
- ic_launcher_round.png, 381
- identyfikator, 382
- JAR, 309, 341
- java.exe, 343
- konfiguracyjny, 303
- module-info.java, 416
- MP3, 296, 298
- plugin.yml, 350, 361
- strings.xml, 381
- styles.xml, 381
- tekstowy odczytywanie, 296
- tworzenie, 294, 314
- tymczasowy, 314

pluskwa, 20

podklasa, 138, 168

pole

- kombi, 248, 249, 278
- tekstowe, 247, 250, 290
- wyboru, 248, 278

polecenie, *Patrz też*: instrukcja, metoda

- java, 55
- Run/Run File, 54
- throw, 201

postfiks, 73

powłoka, 61, 62

prefiks, 73

program, 18, 19, 65, *Patrz też*: aplikacja

- desktopowy, 41
- kompilowanie, 33
- Math, 55
- przyspieszanie, 210
- spowalnianie, 210
- tworzenie, 28
- uruchamianie, 35
- zajętość pamięci, 163

programowanie obiektowe, 48, 133, 134, 169

projekt Spigot, *Patrz*: Spigot

prostokąt, 326

protokół HTTP, 314, 316, 319

przeżytek, 214

przycisk, 262

R

ramka, 241, 266, 323
nazwa, 242
tworzenie, 241
ukrywanie, 243
widoczność, 241
wymiary, 242
zamykanie, 242
Realtime Stock Quotes, 48
REPL, 62

S

SDK Manager, 410
serwer sieciowy, 310
adres, 311
Minecrafta, *Patrz:* Minecraft serwer
serwlet, 41
słowo kluczowe, 70
_, 71
catch, 195
class, 226
extends, 177
finally, 195
for, 110
implements, 211, 276
new, 59, 122
private, 150
protected, 150
public, 136, 150, 158
static, 156
String, 82
super, 168, 175, 385
this, 159, 160, 168, 211, 276
throw, 195
throws, 195
try, 195
void, 91, 153
SourceForge, 49
Spigot, 340, 357
stała, 72
string, *Patrz:* łańcuch
Stroustrup Bjarne, 17
strumień, 293
bajtowy, 301
wejściowy, 294, 295
buforowanie, 298

wyjściowy, 294
znakowy, 294, 301

Ś

środowisko programistyczne zintegrowane,
Patrz: IDE

T

tablica, 121, 171, 180
element
numer, 123
wartość początkowa, 122, 123
sortowanie, 126
tworzenie, 122
wielkość, 122, 124, 131
wielowymiarowa, 125, 126
TLDR, 255
typ
boolean, 69
byte, 68, 307
char, 67, 140, 147
double, 71
float, 67, 71, 72, 140
generyczny, 171, 180
int, 66, 68, 71, 140, 146, 307
konwersja, 139
long, 71, 140
łańcuchowy, 32
short, 68
String, 67, 68, 82, 180
typowanie docelowe, 233

U

unboxing, 141, 187

W

wartość null, 187, 203, 312
wątek, 47, 209
tworzenie, 210
uruchamianie, 217
wykonywanie, 217
zatrzymywanie, 214, 221
wielowątkowość, 47, 209
wiersz poleceń, 21, *Patrz:* konsola

wyjątek, 193, 206, *Patrz też:* błąd
 ArithmeticException, 198
 ArrayIndexOutOfBoundsException, 194
 ArrayIndexOutOfBoundsException, 124
 ignorowanie, 202
 IndexOutOfBoundsException, 181
 InterruptedException, 210, 313
 IOException, 295, 313
 kontrolowany, 202
 MalformedURLException, 202, 205
 niekontrolowany, 202, 203
 NullPointerException, 203
 NumberFormatException, 196, 198
 obsługa, 195
 przechwytywanie, 194, 197, 203
 URISyntaxException, 223, 311
 własny, 206
 zgłaszanie, 194, 200, 203
 wyrażenie, 66, 72, 76
 lambda, 232, 236

Z

Zamenhof Ludwik, 36
 zdarzenie
 klawiatury, 230, 278
 nasłuchiwanie, 275
 obsługa, 218
 pola
 kombi, 278
 wyboru, 278
 źródło, 277
 ziarno, 170
 zmienna, 32, 54
 chroniona, 150, 151
 dekrementacja, 73, 74
 dostęp, 151
 inicjalizowanie, 71, 118
 inkrementacja, 73, 74
 instancyjna, 230
 keyLabel, 230, 231

klasowa, 151, 152, 162
 kontrola dostępu, 150, 151
 łańcuchowa, 82
 nazwa, 68, 70, 79
 obiektowa, 151
 Path, 343
 prosta, 139
 konwersja na obiekt, 141
 prywatna, 150, 151, 154
 publiczna, 150
 rzutowanie, 139, 140
 tworzenie, 66, 150
 typ, 66, *Patrz też:* typ
 wartość początkowa, 71
 zasięg, 156, 157
 znak, 81, 294
 !=, 95
 +=, 85
 ==, 95
 ASCII, 130, 297
 dolara, 70
 kreski pionowej, 198
 nawiasu klamrowego, 31
 podkreślenia, 69, 70, 71
 specjalny, 83
 średnika, 32
 ukośnika, 31
 lewego, 83
 wstecznego, 295
 Unicode, 130, 297
 zachęty jshell>, 62

Ż

żądanie
 GET, 317
 HTTP, 310, 317
 nagłówek, 311
 POST, 316, 317
 sieciowe, 310

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Java to język programowania, którego warto się nauczyć. To technologia nowoczesna i użyteczna, a jej możliwości doceniają największe firmy na całym świecie. Przy tym jest to język dojrzwały i lubiany, wspierany przez rzeszę pasjonatów programowania. Na tym nie koniec zalet Javy: nauka tego języka jest prosta i przyjemna. Aby zacząć pisać aplikacje w Javie, wystarczy odrobina wysiłku i zaangażowania. Oznacza to, że jeśli zechcesz, wkrótce będziesz pisać programy z graficznym interfejsem użytkownika, łączące się z usługami sieciowymi, działające na urządzeniach mobilnych, a nawet kod pracujący w środowisku Minecrafta!

Ta książka to bardzo przystępny, zwięzły podręcznik składający się z 24 godzinnych lekcji. Ich celem jest nauka programowania w języku Java od podstaw — pojęcia programistyczne zatem są objaśniane w sposób zrozumiały, a technika tworzenia kodu została opisana krok po kroku. Po 24 godzinach spędzonych z tą publikacją będziesz pisać własne programy. Nauczysz się projektowania graficznych interfejsów użytkownika, tworzenia aplikacji mobilnych, zrozumiesz zasady programowania obiektowego. Ta książka przygotuje Cię do nauki bardziej zaawansowanych technik programowania w Javie.

Najważniejsze zagadnienia:

- Konfiguracja środowiska programistycznego Javy
- Podstawowe elementy kodu
- Budowanie funkcjonalnych interfejsów użytkownika
- Stosowanie wątków i praca z plikami
- Techniki programowania obiektowego
- Tworzenie aplikacji mobilnych

Rogers Cadenhead jest programistą, twórcą aplikacji, autorem publikacji i wydawcą internetowym. Napisał już ponad 20 książek o tematyce informatycznej. Przez kilka lat był członkiem grupy RSS Advisory Board, która zajmuje się specyfikacją RSS. Swoje artykuły publikuje w Drudge Retort i w innych popularnych serwisach internetowych.



Programowanie w Javie?
Łatwiejsze, niż się wydaje!

SAMS

Helion

Sprawdź nasze szkolenia!

KOD KORZYŚCI
Sięgnij po więcej! ▶



helion.pl

SZKOLENIA



AKADEMIA IT & BUSINESS

ISBN 978-83-283-4235-4



HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

WWW.SZKOLENIA.HELION.PL



INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 69,00 zł